자바 애플릿 기반 원격 제어 모니터 모델 설계 및 구현

이정배[†] · 이종희[††] · 윤석환[†††]

요 약

본 논문에서는 클라이언트/서버 형태로 원격 제어 모니터 모델로서 승용차 조립라인을 시뮬레이션한 컨베이어 시스템을 채택하여 원격으로 제어하는 모니터 모델을 설계하고 구현하였다. 웹 애플릿을 기반으로 구현된 클라이언트 시스템은 상황실에 위치하여 컨베이어 현장을 영상 감시하고 직접적으로 제어를 담당하는 서버에게 컨베이어 제어 명령을 내린다. 제어명령은 서버시스템으로 전달되며, 서버시스템에 존재하는 구동기가 실질적으로 컨베이어를 제어하게 된다.

A Design and Implementation of Remote Control Monitor Model based on Java Applet

Jeong-Bae Lee[†], Jong-Hee Lee[††] and Suk-Hwan Yun[†††]

ABSTRACT

In this paper, client/server model for remote conveyor control and monitoring system is designed and implemented to simulate the product line of vehicles. The client system located at the monitoring office is using a web applet based system, and sends some commands to the server system which guards and controls the conveyer system. The commands is delivered to the server system, the driver software existing in the server controls the conveyer system directly.

1. 서 론

최근에 웹을 기반으로 하여 여러 분야에 적용되는 원격 감시, 제어 시스템 개발에 관한 연구가 활발히 진행되고 있다. 환경 감시, 전력 설비, 무인 공장, 원자력 제어, 보안 시스템과 같은 사람이 현장에서 직접 시스템을 운영하기 어려운 분야에 특성상 컴퓨터의 이용은 필수적이다. 이러한 원격 감시 제어 시스템은 초기에는 중앙집중형을 기반으로 하여 구현되었으나 점차 분산처리를 기반으로 하는 환경으로 급속도로 변환되고 있다. 특히 이러한 분야는 원격으로 관리를 하여야 하는 특수성 때문에 필수적으로 컴퓨터망을 기반으로 하는 것이 기본이다. 최근에는 초고속 정보 통신망의 등장으로 더욱 더 이러한 분야에 원격 감시 및 제어시스템의 도입이 급속도로 이루어지고 있는 추세이다[1,2]. 특히 인터넷이나 인트라넷을 기반으로 웹을 이용하는 시스템 환경 구축에 관한 연구가 활발히 진행되고 있다[3,4]. 웹을 기반으로 하면 데이터베이스와 손쉽게 연동을 할 수 있으면 응용시스템의 개발이 용이하다는 장점을 가지고 있다.

본 연구에서는 그림 1의 목표시스템의 구성도에서 보는 바와 같이 Client/Server 환경에서 초고속정보통신망과 분산 멀티미디어를 이용하여 컨베이어 시스템을 원격으로 제어하는 시스템의 개발을 그 목표로 하고 있다. 즉, 원격지에 있는 컨베이어 시스템을 원격지 현장에서 모든 공정을 통제하는 것과 같은 효과를 상황실 클라이언트 시스템을 통하여 이룰 수 있도록 한다. 이는 일대일 화상문답 채널을 할당함으로써 상황실에서 원격 제어가 가능하도록 한다. 이러한 원격제어는 Inter/Intra-net의 웹을 통하여 가능

† 부산외국어대학교 컴퓨터공학과
†† 신라대학교 컴퓨터정보공학부
††† 정보통신진흥원

그림 1. 목표시스템의 구성도

하도록 한다. 원격제어 시스템은 그림 2에서 보는 바와 같이 클라이언트 측 애플릿 프로그램과 서버측 제어프로그램과 데이터베이스 시스템으로 구성된다. 본 논문에서는 클라이언트측과 제어프로그램이 내장된 서버측과의 원격제어 모니터링 기능을 설명한다.

그림 2. 원격 제어 시스템의 초기 DFD

각 프로세스의 소개에 앞서 원격 제어 시스템의 전체 프로세스 계층도를 살펴보면 그림 3에서 보는 바와 같다. 전체 시스템은 클라이언트와 서버 그리고 DB 관리자로 분류된다. 컨베이어 구동기는 서버관리자 하부의 이벤트 관리자에 의해 구동이 된다.

그림 3. 컨베이어 원격 제어 시스템 프로세스 계층도

2. 클라이언트 시스템

클라이언트 시스템은 원격지 상황실에 존재하며 그림 4와 같이 크게 네트워크 연결관리자, 애플릿 관리자, 이벤트 관리자의 3개의 프로세스로 구성되어 있다.

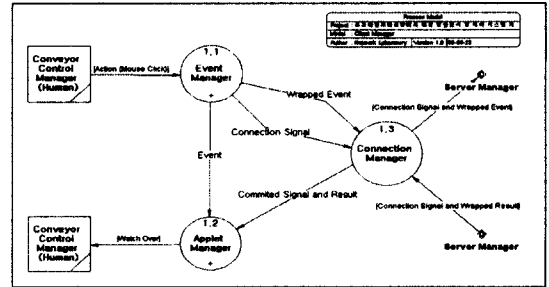



그림 4. 클라이언트 시스템의 DFD

상황실 관리자에 의해 이벤트가 발생하면 네트워크 연결관리자를 통해 현지 공장에 있는 서버 프로세스와 연결하고 주어진 이벤트는 이벤트 관리자에서 포장이 되어 다시 네트워크 연결관리자에 의해 서버 프로세스로 전송되며, 서버 프로세스에서 처리된 결과는 네트워크 연결관리자를 통해 애플릿 관리자로 전송되어 상황실 관리자에서 상황을 보여준다.

3. 서버 시스템

서버 시스템은 원격지 현장에 위치하며 그림 5와 같이 네트워크 연결관리자, DB 관리자, 이벤트 관리

그림 5. 서버 시스템의 DFD

자, 결과 관리자의 4개의 프로세스로 구성되어 있다.

서버 시스템은 클라이언트 시스템으로부터의 이벤트를 분석하여 컨베이어 시스템에 적용시키기도 하며 DB 서버에 질의를 보내기도 한다. 또, 컨베이어 시스템을 체크하여 이벤트 발생시에 DB 서버에 기록하기도 하며 다시 컨베이어 시스템과 클라이언트 시스템으로 현재의 상황을 전송하는 일을 한다.

여기서 서버 시스템의 컨베이어 시스템 모니터링 루틴을 좀 더 자세히 살펴보기로 한다.

우선 본 논문에서 사용되는 컨베이어 시스템은 자동차 조립공정 모델의 실제 구성은 그림 6에서 보는 바와 같다.

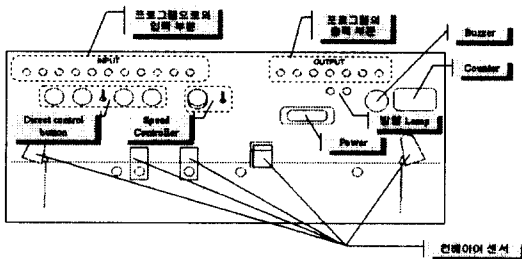



그림 1. 컨베이어 시스템의 구성도

- 프로그램으로의 입력부분 : 컨베이어의 출력으로서 프로그램에 입력되어 현재 컨베이어의 상태를 파악하게 해준다.
- 프로그램의 출력부분 : 프로그램에서 컨베이어 구동기에 의해 구동되는 부분이다.
- Direct control button : 중앙에 있는 auto/man 스위치를 조작하여 man으로 세트된 경우에는 컨베이어 시스템을 현장에서 시스템 관리자가 직접 제어할 수 있도록 해 주며, auto로 세트되는 경우에는 원격으로 컨베이어 시스템을 제어할 수 있는 환경을 제공해 준다.
- Speed Controller : man으로 세트된 경우, 현장에서 컨베이어의 속도를 직접 제어할 수 있도록 한다.
- Power : 컨베이어의 전원 스위치이다.
- 방향 Lamp : 정・역 방향 표시등이다.
- Buzzer : 컨베이어 자체에서 소리를 내는 부분이다.
- Counter : 생산된 승용차의 수를 카운터하는 부분이다.
- 컨베이어 센서 : 각각의 센서는 승용차 생산 라인의 한 공정을 의미한다.

이상의 여러 기능들은 컨베이어 시스템 자체뿐만 아니라 상황실에서 원격 제어 할 수 있는 기능들이다.

컨베이어 원격 제어용 구동기는 서버 시스템에 설치되어 있으며 실질적으로 컨베이어 시스템을 구동하는 함수들을 가지고 있다. 구동기는 Java의 네이티브 메서드로 구현되어 있다.

컨베이어 시스템의 제어는 클라이언트 시스템으로부터 발생되는 이벤트를 체크하는 메인 쓰레드(그림 7)와 컨베이어 시스템의 상황을 주기적으로 체크하는 서브 쓰레드(그림 8)로 구성된다.

```
public static void main(String args[]) {
{
        . . . . .
        while(true) {
            클라이언트 시스템으로부터 시그널을 기다림
            시그널이 오면 이벤트 처리자를 생성
        }
}
```

그림 7. 서버의 메인 쓰레드

서버 시스템의 메인 쓰레드는 예약된 소켓 포트로 클라이언트 시스템으로부터의 접속을 기다리며 접속이 되면 이벤트 관리자를 생성하여 시그널(이벤트)를 처리하게 한다.

```
public void run()
{
        while(true) {
            이전의 컨베이어 시스템의 상황을 저장
            쓰레드를 0.05 초간 쉼
            현재의 컨베이어 시스템의 상황을 체크
            이전의 상황과 현재의 상황이 다를 때 적절
            한 이벤트 처리자를 실행
        }
}
```

그림 8. 서버의 서브 쓰레드

서브 쓰레드는 주기적으로 이전의 컨베이어 시스템의 상황과 현재의 컨베이어 시스템의 상황을 비교

<A Port>
- 서버로 시그널 전송에 사용 (I/O Address 3E8H)

8번 bit	7번 bit	6번 bit	5번 bit	4번 bit	3번 bit	2번 bit	1번 bit
PROX1	LS2	LS1	REV	FOR	AUTO	STOP	START

<B Port>
- 컨베이어 시스템으로 시그널 전송에 사용 (I/O Address 3E9H)

not used	not used	COUNT	BUZZER	REV_L	FOR_L	REV	FOR

<C Port>
- 서버로 시그널 전송과 컨베이어 시스템 속도 설정에 사용
(I/O Address 3EAH)

50%속도	22.5%속도	13.5%속도	6% 속도	not used	not used	PHS	PROX2

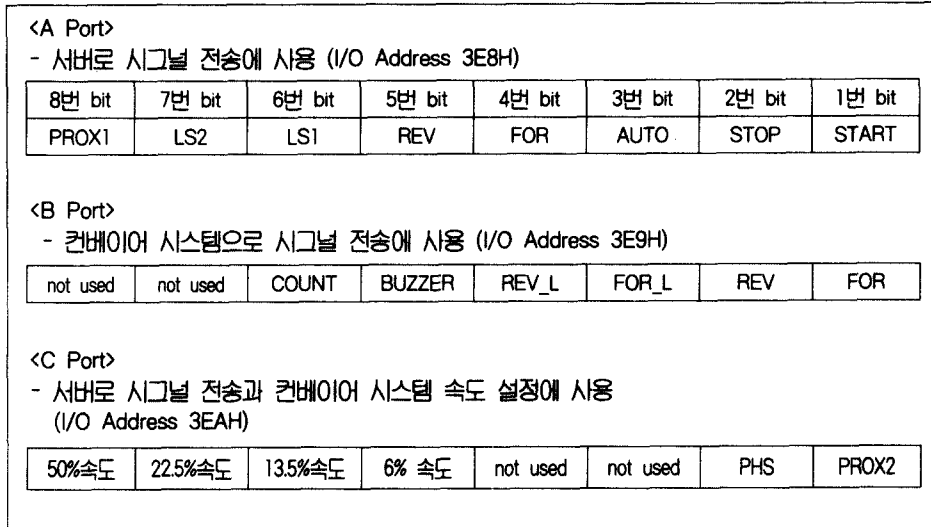



그림 9. A, B, C Port 의 구성도

하여 차이나는 부분에 해당하는 이벤트 관리자를 구동시켜 상황실 관리자 및 현재의 컨베이어 시스템에 적용시키는 일을 수행한다.

컨베이어 시스템은 컨베이어 서버 시스템에 장착된 마이크로 칩을 이용하여 3e8H(A Port), 3e9H(B Port), 3eaH(C Port), 3ebH(Init Port) 의 4개의 I/O address를 사용하여 컨베이어 시스템과 정보를 교환한다. 각각의 I/O address 별로(그림 9) 간단히 설명하면, 3e8H(A Port) address는 컨베이어 시스템의 현재 상황을 서버 시스템으로 전송하는데 사용되며, 3e9H(B Port) address 와 3eaH(C Port) address는 서버의 시그널과 컨베이어 시스템의 속도를 전송하는데 사용된다. 그리고, 3ebH(Init Port) address는 컨베이어 서버 시스템에 장착된 마이크로 칩을 초기화 하는데 사용된다.

먼저 컨베이어 시스템의 초기화(그림 10)는 Init Port에 91H 값을 전송하여 마이크로 칩을 초기화하고 관리자의 입력으로 받아들여진 컨베이어 시스템의 초기 속도값을 셋팅하고 초기에 정해진 방향으로 컨베이어 시스템의 방향을 설정하고 서버로부터 시그널 전송이 이루어질 B Port를 초기화한다.

컨베이어 시스템의 구동(그림 11)은 관리자의 입력으로 받아들여진 컨베이어 시스템의 구동방향을 점검하여 B Port에 그 방향을 전송한다.

이 컨베이어 시스템의 초기 구동 루틴과 아울러

```
void Server__init(struct HServer *this, long Speed)
{
    Init Port에 91H 값을 전송하여 마이크로 칩 초기화
    C Port에 입력된 Speed 값을 전송하여 컨베이어 시
    스템 속도 초기화
    A Port에 움직이는 방향 설정
    B Port에 00H 값을 입력하여 초기화
}
```

그림 10. 컨베이어 시스템 초기화 루틴

```
void Server__act_start(struct HServer *this, long direction)
{
    현재 컨베이어 시스템에 설정된 방향이 정방향이면,
        B Port에 정방향 시그널을 전송하여 구동시킨다.
    현재 컨베이어 시스템에 설정된 방향이 역방향이면,
        B Port에 역방향 시그널을 전송하여 구동시킨다.
}
```

그림 11. 컨베이어 시스템의 초기 구동 루틴

진행 중 속도 증가나 감소 시에도 방향의 전향이 있을 경우에는 그림 12 와 같이 그림 11 과 유사한 루틴을 사용한다. 예를 들면, 컨베이어 시스템의 정방향 진행 시 최소 속도에서 속도 감소 시그널을 만나면 현재의 방향을 체크하여 그와는 반대 방향으로 속도를 증가시켜 주어야 하기 때문이다.

```
void Server__act_start_reverse(struct HServer *this)
{
      B Port의 상태를 읽는다.
      B Port 상태가 정방향이면
         B Port에 역방향 시그널을 전송한다.
      B Port 상태가 역방향이면
         B Port에 정방향 시그널을 전송한다.
}
```

그림 12. 컨베이어 시스템의 방향 전환 루틴

컨베이어 시스템의 정지 루틴(그림 13)은 현재의 상태를 체크하여 구동시에만 B Port에 정지 시그널을 전송하여 컨베이어 시스템을 정지시킨다.

```
void Server__act_stop(struct HServer *this)
{
      현재의 컨베이어 시스템의 상황을 체크한다.
      B Port에 정지 시그널을 전송한다.
}
```

그림 13. 컨베이어 시스템 정지 루틴

그리고, 각 포트의 체크 루틴(그림 14)은 각 포트의 상황을 읽어 그 값을 리턴한다.

```
long Server__check각 포트(struct HServer *this)
{
      A, B, C Port의 상황을 체크한다.
      A, B, C Port의 현재 값을 리턴한다.
}
```

그림 14. A, B, C Port의 체크 루틴

또, 컨베이어 시스템의 속도 제어 루틴은 컨베이어 시스템을 체크하여 총 32 단계로 속도를 제어할 수 있다.

속도 증가 루틴(그림 15)은 현재의 컨베이어 시스템의 방향이 정방향일 때는 최고 속도가 아니라면 한 단계 높은 속도의 시그널을 보내고, 역방향일 때는 정방향으로 진행이 속도의 증가이므로 최저속도가 아니라면 한 단계 낮은 속도를 적용시킴으로써 속도의 향상을 시킨다. 그리고 역방향 최저속도일 때는 방향을 바꾸고 한 단계 높은 속도를 적용시킨다.

```
void Server__SpeedUp(struct HServer *this)
{
      현재의 컨베이어 상황을 체크한다.
      컨베이어 시스템의 방향이 정방향 일때
         최고 속도가 아니면
               한단계 높은 속도를 적용한다.
      컨베이어 시스템의 방향이 역방향 일 때
         최저속도이면
               방향을 바꾸고 속도를 증가시킨다.
         최저속도가 아니면
               한단계 낮은 속도를 적용한다.
}
```

그림 15. 컨베이어 시스템의 속도 증가 루틴

속도 감소 루틴도 증가 루틴(그림 16)과 마찬가지로 컨베이어 시스템의 방향이 정방향 일 때 최저속도가 아니면 한 단계 낮은 속도를 적용시키고 최저속도라면 방향을 바꾸고 속도를 증가시키며 역방향일 때는 최대 속도가 아니면 한 단계 높은 속도를 적용시키고 최대 속도일때는 그대로 둔다.

```
void Server__SpeedDown(struct HServer *this)
{
      현재의 컨베이어 상황을 체크한다.
      컨베이어 시스템의 방향이 정방향 일때
         최저속도이면
               방향을 바꾸고 속도를 증가시킨다.
         최저속도가 아니면
               한단계 낮은 속도를 적용한다.
      컨베이어 시스템의 방향이 역방향 일 때
         최고속도가 아니면
               한단계 높은 속도를 적용한다.
}
```

그림 16. 컨베이어 시스템 속도 감소 루틴

그 외에는 컨베이어 시스템에 부착된 부저를 울리게 하는 루틴(그림 17)과 카운트를 증가시키는 루틴(그림 18)등이 있다.

부저를 울리게 하는 루틴은 일정 시간 동안 B Port에 부저 시그널을 보내어 부저를 울리게 한다.

카운트 제어 루틴은 부저 제어 루틴과 마찬가지로 B Port에 카운트 시그널을 보내어 카운트를 증가시켜 주는 역할을 한다.

이상에서 살펴 본 컨베이어 시스템의 제어 루틴은 최하위 루틴들로써 컨베이어 시스템을 물리적으로

```
void Server__Buzzer(struct HServer *this)
{
    현재의 컨베이어 시스템을 체크한다.
    현재의 컨베이어 시스템의 상황에 부저 시그널을
    OR 시켜 적용한다.
    일정시간 딜레이 지정
    현재의 컨베이어 시스템을 체크한다.
    현재의 컨베이어 시스템의 상황에 부저 시그널을
    빼고 적용한다.
}
```

그림 17. 컨베이어 시스템의 부저 제어 루틴

```
void Server__Counter(struct HServer *this)
{
    현재의 컨베이어 시스템을 체크한다.
    현재의 컨베이어 시스템의 상황에 카운트 시그널
    을 OR 시켜 적용한다.
    일정시간 딜레이 지정
    현재의 컨베이어 시스템을 체크한다.
    현재의 컨베이어 시스템의 상황에 카운트 시그널
    을 빼고 적용한다.
}
```

그림 18. 컨베이어 시스템의 카운트 제어 루틴

제어하며 이를 제어하는 루틴은 Java Application으로서 서버에 존재하여 보다 나은 제어 환경을 유지한다. 또, 현지 공장에서의 직접 제어를 위하여 컨베이어 시스템 직접 제어 스위치를 두어서 원격지 관리자뿐만 아니라 현지 공장의 작업자도 언제든지 컨베이어 시스템을 제어할 수 있다.

4. 시험 및 평가

각 컨베이어 센서에는 조립 중인 승용차들이 있는 것으로 가정한다(그림 19). 센서 1, 2, 3, 4는 그림 6에서 잘 나타나 있다. 그리고 센서 1(내장조립공정)로 조립된 차체(body)가 입력된다고 가정한다. 이 조립된 차체가 센서 1 에 도착하면 일정시간 멈춰서서 공정을 마치게 되며, 이때 데이터베이스에 내장조립공정의 부속들(계기판넬, 히타, 시트, 안전벨트, 에어백)의 재고와 종류를 선택하여 완제품 데이터베이스에 기록되며, 아울러 재고 부품량에 관련된 데이터베이스 내역을 수정하도록 한다. 그리고 센서 2(외장조립공정) 와 3(파워트레인공정) 역시 앞의 센서와 동일한 작업을 수행한다. 그리고 센서 3 을 지나게 되면 승용차를 검사하여 출고하는 공정으로 넘어가게 된다. 이때 센서 4(불량품감지센서)까지 도달하는 승용차는 불량품이라 보고 데이터베이스에 기록을 하게 된다. 그리고 승용차의 고유번호로써 검색이 가능하며 불량품 승용차에 들어간 부품들의 명세도 볼 수 있게 된다.

본 연구에서는 승용차들을 가정한 물체를 10개 통과시키고 그 중에 5개는 불량 조립 승용차로 가정한 물체를 통과시켜서 위에서 설명한 평가 방법으로 컨베이어 시스템의 동작을 평가하였다. 그 결과평가표는 그림 19 에서 보는 바와 같이 각 공정에서 정상적으로 동작함을 알 수 있었으며, 불량품의 식별 공정도 정상적으로 수행됨을 알 수 있었다.

	내장조립 센서 1	외장조립 센서 2	파워트레인 센서 3	불량품감지 센서 4	비고
정상 공정	정상 공정 확인	정상 공정 확인	정상 공정 확인	정상 공정 확인	
	내장조립공정 DB에서 각 key를 뽑아 완제품 DB에 넣는다.	외장조립공정 DB에서 각 key를 뽑아 완제품 DB에 넣는다.	파워트레인공정 DB에서 각 key를 뽑아 완제품 DB에 넣는다.	센서3을 통과 후 일정시간내에 다음공정으로 넘어가게 되면 정상제품으로 표시한다.	
비정상 공정	비정상공정확인	비정상공정확인	비정상공정확인	불량품 감지	
	내장조립공정 DB에서 각 key를 뽑아 완제품 DB에 넣는다.	외장조립공정 DB에서 각 key를 뽑아 완제품 DB에 넣는다.	파워트레인공정 DB에서 각 key를 뽑아 완제품 DB에 넣는다.	현 완제품 승용차 DB에 불량품으로 표시한다.	

그림 19. 결과 평가표

5. 결 론

본 연구에서는 클라이언트/서버 형태로 Java 네이티브 메서드를 이용하여 컨베이어 시스템을 원격으로 제어하고 모니터링하는 방법을 제시하고 구현하였다. 이러한 시스템의 구성은 근거리통신망으로 원격지에 연결된 컨베이어 시스템을 상황실에 위치한 클라이언트 시스템에서 웹 애플릿을 사용자 인터페이스로 사용하여 원격 제어 및 모니터링하는 형태를 가진다. 또, 서버 시스템에서는 컨베이어 시스템을 통과하는 생산품 내역에 대한 데이터베이스를 저장, 유지, 관리한다. 본 연구에서 사용된 시스템은 승용차 조립라인을 모델링한 시스템으로 구성되어있다. 현재 원격제어 및 모니터링 대상은 승용차 조립라인의 컨베이어 시스템을 1차 목표로 하여 구현하였다. 장기적으로는 승용차 조립라인의 모든 제어기들과 연동될 수 있도록 한다.

이러한 자동화 기술이 성공적으로 이루어지므로써 초고속정보통신망에서 분산 멀티미디어 사용 기술의 확산은 더욱 더 활발해 질 것이며, 원격 제어 및 모니터링뿐만 아니라 물류 및 통합 공정시스템등 다양한 분야에 활용할 수 있는 기술의 개발이 가능해진다. 원격지 공정과 상황실 시스템간의 편리한 대화를 위해서는 텍스트, 그래픽, 이미지, 사운드, 비디오 등 멀티 미디어에 의한 대화가 가능해야 하는데, 이 시스템의 개발로 말미암아 다양한 미디어에 의한 인터페이스 개발의 가속화를 기대할 수 있게 된다. 하이퍼미디어에 바탕을 둔 사용자 인터페이스에 관한 연구는 객체 지향 기술, 시각프로그래밍, 멀티미디어 기술 등 여러 분야에 파급되어 편리한 사용자 환경 구축을 통해 전문가가 아닌 초심자들도 쉽게 조작할 수 있을 것이다. 공정 제어 분야에서는 공장 운영에 상당한 비용 절감을 기대해 볼 수 있고, 생산성 향상을 꾀할 수 있을 것으로 기대가 되므로 초고속정보통신망 하에서 원격 영상감시 및 제어 시스템의 시장성은 대단하다고 볼 수 있고, 전체 산업 발전에 끼치는 영향은 크다고 할 것이다.

초고속정보통신망에서 분산 멀티미디어 기술을 기반으로 한 이 자동화 기술은 원격 영상 감시, 제어뿐만 아니라 물류 및 통합 공정시스템등 다양한 분야에 활용할 수 있는 기술 개발이 가능하다 향후 실시간성을 보장하는 분산 데이터 베이스 시스템과 연계하여 통합 물류 및 공정 제어 시스템에의 활용을 도모하고자 한다. 또한 실시간 시스템으로 전송 프로토콜을 강화하여 공장자동화, 물류 자동화 등에 적용할 수 있는 시스템으로 개선되어 갈 것으로 기대되며, 국내 산학연간 협동 연구를 통하여 생산공정에 관한 기술적인 협력을 도모하여 실용 방안의 모색이 절실히 요구된다.

참 고 문 헌

[1] Craig M. Wittenbrick, Eric C. Rosen, Darrell D. E. Long, "Real-time System for Managing Environmental Data." Proceeding of Conference on Software Engineering and Knowledge Engineering, June 1996

[2] Theodore R. Haining, Darrell D. E. Long, Patric E. Mantey, Craig M. Wittenbrink, "The Real-Time Environmental Information Network and Analysis System(REINAS), "Proceeding of COMPCON, March 1995

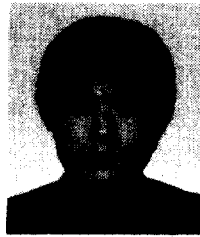
[3] 이 정배, 김 인홍, "원격 영상 감시 및 제어 자동화," 정보처리학회지, 1997. 7.

[4] 이 정배, "웹을 기반으로 한 원격 UNIX 관리 시스템 구현에 관한 연구", 1998년 정보처리학회 추계학술발표대회, 1998. 10.

이 정 배

1981년 경북대학교 전자공학과 전산전공 공학사
1983년 경북대학교 전자공학과 전산전공 공학석사
1995년 한양대학교 전자공학과 공학박사
1982년~1991년 한국전자통신연구소(ETRI) 선임연구원
1996년~1997년 U.C.Irvine 개원교수
1991년~현재 부산외국어대학교 부교수
관심분야 : 컴퓨터 네트워크, 실시간 자바, 실시간 프로토콜, 인터넷 응용, 컴퓨터교육

이 종 희

1978년 경북대학교 공과대학 전자공학과 졸업(공학사)
1984년 경북대학교 대학원 전자공학과 전산공학 졸업(공학석사)
1990년 경북대학교 대학원 전자공학과 전산공학전공 졸업(공학박사)
1979년~1987년 경남정보대학 조교수
1988년~현재 신라대학교 컴퓨터정보공학부 교수
관심분야 : 인터넷응용, 퍼지신경망, 컴퓨터교육
e-mail : jhlee@silla.ac.kr

윤 석 환

1982년 2월 아주대 공과대 산업공학과 졸업(공학사)
1984년 2월 건국대 대학원 산업공학과 졸업(공학석사)
1992년 8월 품질관리 기술사 자격취득(등록번호 : 92137010342V)
1996년 8월 아주대 대학원 산업공학과 졸업(공학박사)
1986년 1월 한국전자통신연구원 책임연구원(컴퓨터연구단)
1998년 1월 현재 정보통신연구진흥원 책임연구원(융자1팀장)
한국정보처리학회 학회지 편집위원장('98.1. ?현재)